

# MIPS-C 指令集

(版本：1.0)

文档编号：COCO01

高小鹏  
北京航空航天大学计算机学院  
2014 年

## 修订记录

正式发布 1.0 版。

## 目录

A.1 MIPS-C 指令表 .....	5
A.2 MIPS-C 指令图 .....	7
A.3 指令详解(按字母排列).....	8
1. ADD: 符号加 .....	8
2. ADDI: 符号加立即数 .....	8
3. ADDIU: 无符号加立即数 .....	8
4. ADDU: 无符号加 .....	9
5. AND: 与 .....	9
6. ANDI: 与立即数 .....	9
7. BEQ: 相等时转移 .....	10
8. BGEZ: 大于等于 0 时转移 .....	10
9. BGTZ: 大于 0 时转移 .....	10
10. BLEZ: 小于等于 0 时转移 .....	10
11. BLTZ: 小于 0 时转移 .....	11
12. BNE: 不等于时转移 .....	11
13. BREAK: 断点 .....	11
14. DIV: 符号除 .....	12
15. DIVU: 无符号除 .....	12
16. ERET: 异常返回 .....	12
17. J: 跳转 .....	13
18. JAL: 跳转并链接 .....	13
19. JALR: 跳转并链接 .....	13
20. JR: 跳转至寄存器 .....	14
21. LB: 加载字节 .....	14
22. LBU: 加载无符号字节 .....	14
23. LH: 加载半字 .....	14

24.	LHU: 加载无符号半字 .....	15
25.	LUI: 立即数加载至高位 .....	15
26.	LW: 加载字 .....	15
27.	MFC0: 读 CP0 寄存器 .....	16
28.	MFHI: 读 HI 寄存器 .....	16
29.	MFLO: 读 LO 寄存器 .....	16
30.	MTC0: 写 CP0 寄存器 .....	16
31.	MTHI: 写 HI 寄存器 .....	17
32.	MTLO: 写 LO 寄存器 .....	17
33.	MULT: 符号乘 .....	17
34.	MULTU: 无符号乘 .....	18
35.	NOR: 或非 .....	18
36.	OR: 或 .....	18
37.	ORI: 或立即数 .....	19
38.	SB: 存储字节 .....	19
39.	SH: 存储半字节 .....	19
40.	SLL: 逻辑左移 .....	19
41.	SLLV: 逻辑可变左移 .....	20
42.	SLT: 小于置 1(有符号) .....	20
43.	SLTI: 小于立即数置 1(有符号) .....	20
44.	SLTIU: 小于立即数置 1(无符号) .....	21
45.	SLTU: 小于置 1(无符号) .....	21
46.	SRA: 算术右移 .....	21
47.	SRAV: 算术可变右移 .....	21
48.	SRL: 逻辑右移 .....	22
49.	SRLV: 逻辑可变右移 .....	22
50.	SUB: 符号减 .....	22
51.	SUBU: 无符号减 .....	23
52.	SW: 存储字 .....	23
53.	SYSCALL: 系统调用 .....	23

54.	XOR: 异或.....	24
55.	XORI: 异或立即数 .....	24

## A.1 MIPS-C 指令表

本附录从 MIPS32 指令集中选择了一些常用指令构成了 MIPS-C 指令集。MIPS-C 可以支持除浮点运算外的绝大多数定点类程序的运行，并且提供了包括 CP0、异常处理等指令，可以支持简单的操作系统的运行。MIPS-C 指令集共包括 55 条指令。从更细致的功能角度，MIPS-C 被划分为 9 个子类。

功能分类	助记符	功能	OPCODE/ FUNCT (16 进制)	操作 (VerilogHDL 语法描述)
加载	LB	加载字节	20H	$R[rt] = \{24\{\text{Mem}[GPR[rs] + \text{sign\_ext}(\text{offset})][7]\}, \text{Mem}[GPR[rs] + \text{sign\_ext}(\text{offset})][7:0]\}$
	LBU	加载字节 (无符号)	24H	$R[rt] = \{24'b0, \text{Mem}[GPR[rs] + \text{sign\_ext}(\text{offset})][7:0]\}$
	LH	加载半字	21H	$R[rt] = \{16\{\text{Mem}[GPR[rs] + \text{sign\_ext}(\text{offset})][15]\}, \text{Mem}[GPR[rs] + \text{sign\_ext}(\text{offset})][15:0]\}$
	LHU	加载半字 (无符号)	25H	$R[rt] = \{16'b0, \text{Mem}[GPR[rs] + \text{sign\_ext}(\text{offset})][15:0]\}$
	LW	加载字	23H	$R[rt] = \text{Mem}[GPR[rs] + \text{sign\_ext}(\text{offset})]$
保存	SB	存储字节	28H	$\text{Mem}[GPR[rs] + \text{sign\_ext}(\text{offset})][7:0] = R[rt][7:0]$
	SH	存储半字	29H	$\text{Mem}[GPR[rs] + \text{sign\_ext}(\text{offset})][15:0] = R[rt][15:0]$
	SW	存储字	2BH	$\text{Mem}[GPR[rs] + \text{sign\_ext}(\text{offset})] = R[rt]$
R-R 运算	ADD	加	0/20H	$GPR[rd] = GPR[rs] + GPR[rt]$
	ADDU	无符号加	0/21H	$GPR[rd] = GPR[rs] + GPR[rt]$
	SUB	减	0/22H	$GPR[rd] = GPR[rs] - GPR[rt]$
	SUBU	无符号减	0/23H	$GPR[rd] = GPR[rs] - GPR[rt]$
	MULT	乘	0/18H	$\{HI, LO\} = GPR[rs] \times GPR[rt]$
	MULTU	乘(无符号)	0/19H	$\{HI, LO\} = GPR[rs] \times GPR[rt]$
	DIV	除	0/1AH	$\{HI, LO\} = GPR[rs] / GPR[rt]$
	DIVU	除(无符号)	0/1BH	$\{HI, LO\} = GPR[rs] / GPR[rt]$
	SLT	小于置 1	0/2AH	$GPR[rd] = (GPR[rs] < GPR[rt]) ? 1:0$
	SLTU	小于置 1 (无符号)	0/2BH	$GPR[rd] = (GPR[rs] < GPR[rt]) ? 1:0$
	SLL	逻辑左移	0/0H	$GPR[rd] = \{GPR[rt][31-s:0], s\{0\}\}$
	SRL	逻辑右移	0/2H	$GPR[rd] = \{s\{0\}, GPR[rt][31:s]\}$
	SRA	算术右移	0/3H	$GPR[rd] = \{s\{GPR[rt][31]\}, GPR[rt][31:s]\}$
	SLLV	逻辑可变左移	0/4H	$GPR[rd] = \{GPR[rt][31-v:0], v\{0\}\}$
	SRLV	逻辑可变右移	0/6H	$GPR[rd] = \{v\{0\}, GPR[rt][31:v]\}$
	SRAV	算术可变右移	0/7H	$GPR[rd] = \{v\{GPR[rt][31]\}, GPR[rt][31:v]\}$
	AND	与	0/24H	$GPR[rd] = GPR[rs] \& GPR[rt]$
	OR	或	0/25H	$GPR[rd] = GPR[rs]   GPR[rt]$
	XOR	异或	0/26H	$GPR[rd] = GPR[rs] \wedge GPR[rt]$
	NOR	或非	0/27H	$GPR[rd] = \sim(GPR[rs]   GPR[rt])$
R-I	ADDI	加立即数	8H	$GPR[rt] = GPR[rs] + \text{SignExt}(\text{Imm})$

运算	ADDIU	加立即数 (无符号)	9H	$GPR[rt] = GPR[rs] + \text{SignExt}(Imm)$
	ANDI	与立即数	CH	$GPR[rt] = GPR[rs] \& \text{ZeroExt}(Imm)$
	ORI	或立即数	DH	$GPR[rt] = GPR[rs]   \text{ZeroExt}(Imm)$
	XORI	异或立即数	EH	$GPR[rt] = GPR[rs] \wedge \text{ZeroExt}(Imm)$
	LUI	立即数加载至高 位	FH	$GPR[rt] = \{imm, 16'b0\}$
	SLTI	小于立即数置 1	AH	$GPR[rt] = (GPR[rs] < \text{SignExt}(Imm)) ? 1 : 0$
	SLTIU	小于立即数置 1 (无符号)	BH	$GPR[rt] = (GPR[rs] < \text{SignExt}(Imm)) ? 1 : 0$
分支	BEQ	等于转移	4H	if ( $GPR[rs] == GPR[rt]$ ) PC = PC + 4 + BranchAddr
	BNE	不等转移	5H	if ( $GPR[rs] != GPR[rt]$ ) PC = PC + 4 + BranchAddr
	BLEZ	小于等于 0 转移	6H	if ( $GPR[rs] \leq 0$ ) PC = PC + 4 + BranchAddr
	BGTZ	大于 0 转移	7H	if ( $GPR[rs] > 0$ ) PC = PC + 4 + BranchAddr
	BLTZ	小于 0 转移	特殊编码①	if ( $GPR[rs] < 0$ ) PC = PC + 4 + BranchAddr
	BGEZ	大于等于 0 转移	特殊编码②	if ( $GPR[rs] \geq 0$ ) PC = PC + 4 + BranchAddr
跳转	J	跳转	2H	PC = JumpAddr
	JAL	跳转并链接	3H	PC = JumpAddr; $GPR[31] = PC + 4$
	JALR	跳转并链接寄存 器	0/9H	PC = $GPR[rs]$ ; $GPR[rd] = PC + 4$
	JR	跳转寄存器	0/8H	PC = $GPR[rs]$
传输	MFHI	读 HI 寄存器	0/10H	$GPR[rd] = HI$
	MFLO	读 LO 寄存器	0/12H	$GPR[rd] = LO$
	MTHI	写 HI 寄存器	0/11H	$HI = GPR[rs]$
	MTLO	写 LO 寄存器	0/13H	$LO = GPR[rs]$
特权	ERET	异常返回	10/18H	PC = EPC; 还需要对 CP0 的其他寄存器做处理
	MFC0	读 CP0 寄存器	特殊编码③	$GPR[rt] = CP0[rd]$
	MTC0	写 CP0 寄存器	特殊编码④	$CP0[rd] = GPR[rt]$
陷阱	BREAK	断点异常	0/DH	EPC = PC+4; PC = 异常处理地址; CP0 的其他寄存器做处理
	SYSCALL	系统调用异常	0/CH	EPC = PC+4; PC = 异常处理地址; CP0 的其他寄存器做处理

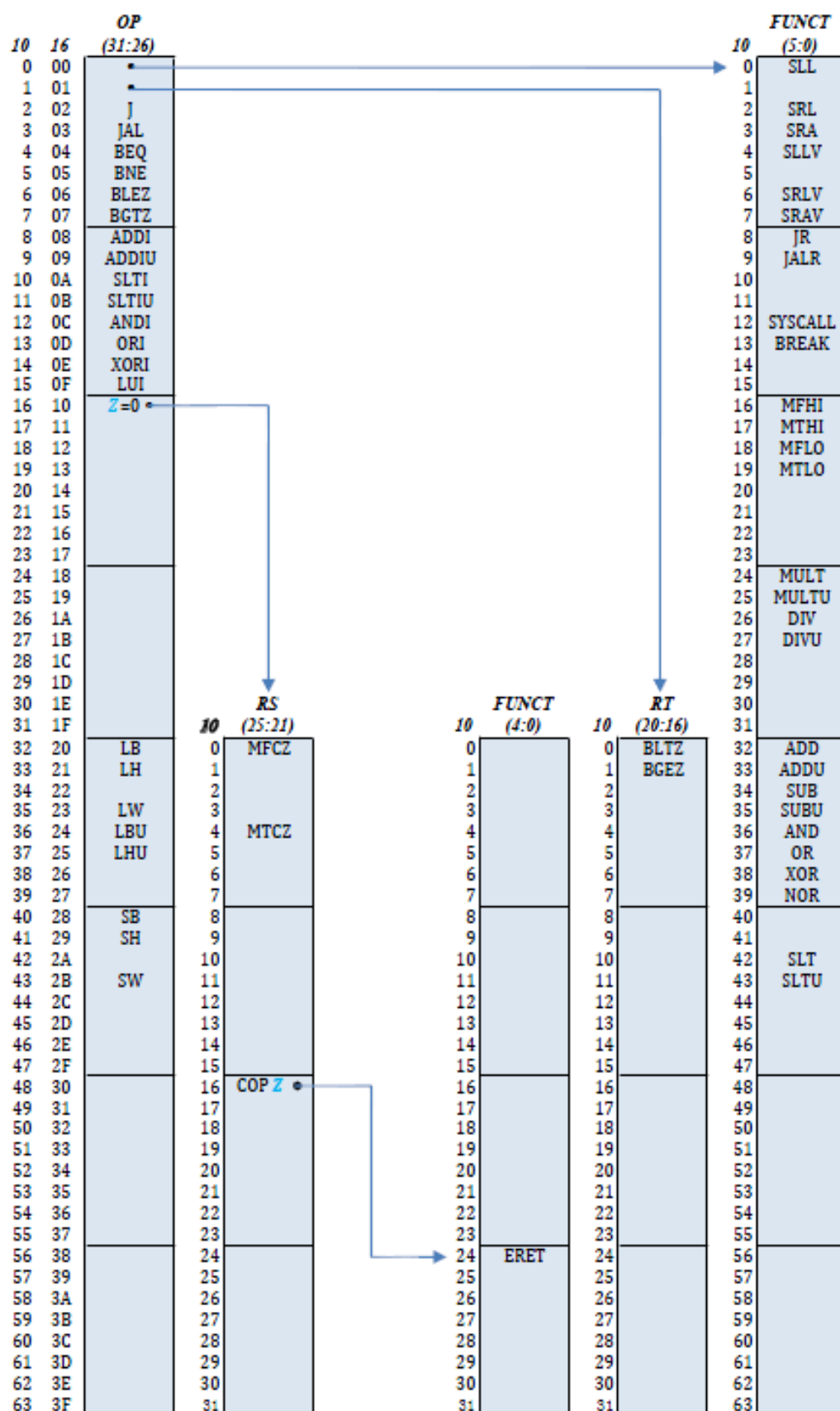
①BLTZ:  $\text{INSTR}_{31..26} / \text{INSTR}_{20..16} = 01/00H$

②BGEZ:  $\text{INSTR}_{31..26} / \text{INSTR}_{20..16} = 01/01H$

③MFC0:  $\text{INSTR}_{31..26} / \text{INSTR}_{25..21} = 10/00H$

④MTC0:  $\text{INSTR}_{31..26} / \text{INSTR}_{25..21} = 10/04H$

## A.2 MIPS-C 指令图



## A.3 指令详解(按字母排列)

55 条 MIPS-C 指令按字母排序。每条指令均包括指令编码(encoding)、格式(format)、描述(description)、操作(operation)、示例(example)和其他(note)。其中最为重要的描述和操作部分。描述部分用 RTL(Register Transfer Language)方式定义了指令的基本操作语义，操作部分则用 RTL 定义了指令的详细操作语义。

### 1. ADD: 符号加

编码	31	26	25	21	20	16	15	11	10	6	5	0
	special 000000	rs		rt		rd		0 00000		add 100000		
	6	5		5		5		5		6		
格式	add rd, rs, rt											
描述	GPR[rd] ← GPR[rs]+GPR[rt]											
操作	temp ← (GPR[rs] <sub>31</sub>   GPR[rs]) + (GPR[rt] <sub>31</sub>   GPR[rt]) if temp <sub>32</sub> ≠ temp <sub>31</sub> then SignalException(IntegerOverflow) else GPR[rd] ← temp <sub>31..0</sub> endif											
示例	add \$s1, \$s2, \$s3											
其他	temp <sub>32</sub> ≠ temp <sub>31</sub> 代表计算结果溢出。 如果不考虑溢出，则 add 与 addu 等价。											

### 2. ADDI: 符号加立即数

编码	31		26		25		21		20		16		15		0	
	addi 001000				rs				rt				immediate			
	6				5				5				16			
格式	addi rt, rs, immediate															
描述	GPR[rt] ← GPR[rs]+ immediate															
操作	temp ← (GPR[rs] <sub>31</sub>   GPR[rs]) + sign_extend(immediate) if temp <sub>32</sub> ≠ temp <sub>31</sub> then SignalException(IntegerOverflow) else GPR[rt] ← temp <sub>31..0</sub> endif															
示例	addi \$s1, \$s2, -1															
其他	temp <sub>32</sub> ≠ temp <sub>31</sub> 代表计算结果溢出。 如果不考虑溢出，则 addi 与 addiu 等价。															

### 3. ADDIU: 无符号加立即数

编码	31	26	25	21	20	16	15	0
	addiu 001001		rs		rt		immediate	





7. BEQ: 相等时转移

编码	312625212016150	
	beq 000100	rsrtoffset
	65516	
格式	beq rs, rt, offset	
描述	if (GPR[rs] == GPR[rt]) then 转移	
操作	if (GPR[rs] == GPR[rt]) PC ← PC + 4 + sign_extend(offset  0 <sup>2</sup> ) else PC ← PC + 4	
示例	beq \$s1, \$s2, -2	
其他		

8. BGEZ: 大于等于 0 时转移

编码	31	26	25	21	20	16	15	0
	000001		rs		bgez 00001		offset	
	6		5		5		16	
格式	bgez rs, offset							
描述	if (GPR[rs] >= 0) then 转移							
操作	if (GPR[rs] >= 0) PC ← PC + 4 + sign_extend(offset  0 <sup>2</sup> ) else PC ← PC + 4							
示例	bgez \$s1, -2							
其他								

9. BGTZ: 大于 0 时转移

编码	31	26	25	21	20	16	15	0
	bgtz 000111		rs		0 00000		offset	
	6		5		5		16	
格式	bgtz rs, offset							
描述	if (GPR[rs] > 0) then 转移							
操作	if (GPR[rs] > 0) PC ← PC + 4 + sign_extend(offset  0 <sup>2</sup> ) else PC ← PC + 4							
示例	bgtz \$s1, -2							
其他								

10. BLEZ: 小于等于 0 时转移

编码	31	26	25	21	20	16	15	0
----	----	----	----	----	----	----	----	---

	blez 000110	rs	0 00000	offset
	6	5	5	16
格式	blez rs, offset			
描述	if (GPR[rs] <= 0) then 转移			
操作	if (GPR[rs] <= 0) PC ← PC + 4 + sign_extend(offset  0 <sup>2</sup> ) else PC ← PC + 4			
示例	blez \$s1, -2			
其他				

## 11. BLTZ: 小于 0 时转移

编码	31	26	25	21	20	16	15	0
	000001	rs	bltz 00000	offset				
	6	5	5	16				
格式	bltz rs, offset							
描述	if (GPR[rs] < 0) then 转移							
操作	if (GPR[rs] < 0) PC ← PC + 4 + sign_extend(offset  0 <sup>2</sup> ) else PC ← PC + 4							
示例	bltz \$s1, -2							
其他								

## 12. BNE: 不等于时转移

编码	31	26	25	21	20	16	15	0
	bne 000101	rs	rt	offset				
	6	5	5	16				
格式	bne rs, rt, offset							
描述	if (GPR[rs] ≠ GPR[rt]) then 转移							
操作	if (GPR[rs] ≠ GPR[rt]) PC ← PC + 4 + sign_extend(offset  0 <sup>2</sup> ) else PC ← PC + 4							
示例	bne \$s1, \$s2, 8							
其他								

## 13. BREAK: 断点

编码	31	26	25	6	5	0
	SPECIAL 000000	code				BREAK 001101
	6	20				6





其他	jalr 与 jr 配套使用。jal 用于调用函数，jr 用于函数返回。
----	--------------------------------------

## 20. JR: 跳转至寄存器

编码	31		26		25		21		20		11		10		6		5		0	
	special 000000				rs				0 00 0000 0000				0 00000				jr 001000			
	6				5				10				5				6			
	格式																			
jr rs																				
描述																				
PC ← GPR[rs]																				
操作																				
PC ← GPR[rs]																				
示例																				
jr \$31																				
其他																				
jr 与 jal/jalr 配套使用。jal/jalr 用于调用函数，jr 用于函数返回。																				

## 21. LB: 加载字节

编码	31	26	25	21	20	16	15	0
	lb 100000		base		rt		offset	
	6		5		5		16	
格式	lb rt, offset(base)							
描述	$GPR[rt] \leftarrow memory[GPR[base]+offset]$							
操作	Addr $\leftarrow GPR[base] + sign\_ext(offset)$ memword $\leftarrow memory[Addr]$ byte $\leftarrow Addr_{1..0}$ $GPR[rt] \leftarrow sign\_ext(memword_{7+8*byte..8*byte})$							
示例	lb \$v1, 3(\$s0)							

## 22. LBU: 加载无符号字节

编码	31	26	25	21	20	16	15	0
	lbu 100100		base		rt		offset	
	6		5		5		16	
格式	lbu rt, offset(base)							
描述	$GPR[rt] \leftarrow memory[GPR[base]+offset]$							
操作	$Addr \leftarrow GPR[base] + sign\_ext(offset)$							
	$memword \leftarrow memory[Addr]$ $byte \leftarrow Addr_{1..0}$ $GPR[rt] \leftarrow zero\_ext(memword_{7+8*byte..8*byte})$							
示例	lbu \$v1, 3(\$s0)							

## 23. LH: 加载半字

编码	31	26	25	21	20	16	15	0
	lh 100001		base		rt		offset	



	$\text{GPR}[\text{rt}] \leftarrow \text{memory}[\text{Addr}]$
示例	<code>lw \$v1, 8(\$s0)</code>
约束	Addr 必须是 4 的倍数(即 $\text{Addr}_{1..0}$ 必须为 00)，否则产生地址错误异常

## 27. MFC0: 读 CP0 寄存器

编码	31	26	25	21	20	16	15	11	10	0
	COP0 010000		mfc0 00000		rt	rd		0 0 0000 0000		
	6		5		5	5		11		
格式	mfc0 rt, rd									
描述	GPR[rt] ← CP0[rd]									
操作	GPR[rt] ← CP0[rd]									
示例	mfc0 \$s1, \$1									
其他										

## 28. MFHI: 读 HI 寄存器

编码	31	26	25	21	20	16	15	11	10	6	5	0
	special 000000		0 00 0000 0000				rd		0 00000		mfhi 010000	
	6		10				5		5		6	
格式	mfhi rd											
描述	GPR[rd] ← HI											
操作	GPR[rd] ← HI											
示例	mfhi \$s1											
其他	当乘法/除法计算完毕后，需要用 mfhi 读取相应的结果。											

## 29. MFLO: 读 LO 寄存器

编 码	31	26	25	21	20	16	15	11	10	6	5	0
	special 000000		0 00 0000 0000				rd		0 00000		mflo 010010	
	6		10				5		5		6	
格 式	mflo rd											
描 述	GPR[rd] ← LO											
操 作	GPR[rd] ← LO											
示 例	mflo \$s1											
其 他	当乘法/除法计算完毕后，需要用 mflo 读取相应的结果。											

## 30. MTC0: 写 CP0 寄存器







示例	or \$s1, \$s2, \$s3
其他	

### 37. ORI: 或立即数

编码	31	26	25	21	20	16	15	0
	ori 001101		rs		rt		immediate	
	6		5		5		16	
格式	ori rt, rs, immediate							
描述	GPR[rt] ← GPR[rs] OR immediate							
操作	GPR[rt] ← GPR[rs] OR zero_extend(immediate)							
示例	ori \$s1, \$s2, 0x55AA							
其他								

### 38. SB: 存储字节

编码	31	26	25	21	20	16	15	0
	sb 101000		base		rt		offset	
	6		5		5		16	
格式	sb rt, offset(base)							
描述	memory[GPR[base]+offset] ← GPR[rt]							
操作	Addr ← GPR[base] + sign_extend(offset) byte← Addr <sub>1..0</sub> memory[Addr] <sub>7+8*byte..8*byte</sub> ← GPR[rt] <sub>7:0</sub>							
示例	sb \$v1, 3(\$s0)							

### 39. SH: 存储半字节

编码	31	26	25	21	20	16	15	0
	sh 101001		base		rt		offset	
	6		5		5		16	
格式	sh rt, offset(base)							
描述	memory[GPR[base]+offset] ← GPR[rt]							
操作	Addr ← GPR[base] + sign_extend(offset) byte ← Addr <sub>1</sub> memory[Addr] <sub>15+16*byte..16*byte</sub> ← GPR[rt] <sub>15:0</sub>							
示例	sh \$v1, 24(\$s0)							
约束	Addr 必须是 2 的倍数(即 Addr <sub>0</sub> 必须为 0), 否则产生地址错误异常							

### 40. SLL: 逻辑左移

编码	31	26	25	21	20	16	15	11	10	6	5	0
----	----	----	----	----	----	----	----	----	----	---	---	---



示例	slti \$s1, \$s2, 0x55AA
其他	

#### 44. SLTIU: 小于立即数置 1(无符号)

编码	31	26	25	21	20	16	15	0		
	sltiu 001011		rs		rt		immediate			
	6		5		5		16			
格式	sltiu rt, rs, immediate									
描述	$\text{GPR}[\text{rt}] \leftarrow (\text{GPR}[\text{rs}] < \text{immediate})$									
操作	$\text{GPR}[\text{rt}] \leftarrow (0 \parallel \text{GPR}[\text{rs}] < 0 \parallel \text{sign\_extend}(\text{immediate})) ? 0^{31} \parallel 1 : 0^{32}$									
示例	sltiu \$s1, \$s2, 0xAABB									
其他	“无符号” 是误导									

#### 45. SLTU: 小于置 1(无符号)

编码	31	26	25	21	20	16	15	11	10	6	5	0
	special 000000		rs		rt		rd		00000		sltu 101011	
	6		00000		5		5		5		6	
格式	sltu rd, rs, rt											
描述	GPR[rd] ← (GPR[rs] < GPR[rt])											
操作	GPR[rd] ← (0  GPR[rs]< 0  GPR[rt]) ? 0 <sup>31</sup>   1 : 0 <sup>32</sup>											
示例	sltu \$s1, \$s2, \$s3											
其他												

#### 46. SRA: 算术右移

编码	31	26	25	21	20	16	15	11	10	6	5	0
	special 000000		0		rt		rd		s		sra 000011	
	6		00000		5		5		5		6	
格式	sra rd, rt, s											
描述	GPR[rd] ← GPR[rt] >> s											
操作	GPR[rd] ← GPR[rt] <sub>31</sub> <sup>*</sup>    GPR[rt] <sub>31..s</sub>											
示例	sra \$s1, \$s2, 5											
其他												

#### 47. SRAV: 算术可变右移

编码	31	26	25	21	20	16	15	11	10	6	5	0
	special		rs		rt		rd		00000		srav	





